Theoretical Computer Science 519 (2014) 29–45

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Combining initial segments of lists ☆

Manfred K. Warmuth [a,1], Wouter M. Koolen [b,c,2], David P. Helmbold [a]

[a] Department of Computer Science, UC Santa Cruz, United States
[b] Department of Computer Science, Royal Holloway, University of London, Egham, Surrey TW200EX, United Kingdom
[c] Centrum Wiskunde en Informatica (CWI), P.O. Box 94079, 1090 GB Amsterdam, The Netherlands

ARTICLE INFO

Keywords:
Online learning
Ranking
Worst-case analysis
Relative loss bounds

ABSTRACT

We propose a new way to build a combined list from K base lists, each containing N items. A combined list consists of top segments of various sizes from each base list so that the total size of all top segments equals N. A sequence of item requests is processed and the goal is to minimize the total number of misses. That is, we seek to build a combined list that contains all the frequently requested items. We first consider the special case of disjoint base lists. There, we design an efficient algorithm that computes the best combined list for a given sequence of requests. In addition, we develop a randomized online algorithm whose expected number of misses is close to that of the best combined list chosen in hindsight. We prove lower bounds that show that the expected number of misses of our randomized algorithm is close to the optimum. In the presence of duplicate items, we show that computing the best combined list is NP-hard. We show that our algorithms still apply to a linearized notion of loss in this case. We expect that this new way of aggregating lists will find many ranking applications.

1. Introduction

We propose a new approach for aggregating ranked lists. Assume we have K lists of N items each, as illustrated in Fig. 1. Our comparator is the best combined list of size N that is composed of the tops of the K lists. A combined list might take the top 20% of list 1, the top 0% of list 2, the top 60% of list 3, and so forth. Note that the contents of the base lists might change over time, and there are exponentially many (roughly N^K) combined lists altogether.

We seek efficient online algorithms that construct such combined lists on the fly. In each trial the following happens: First, the current contents of all base lists are provided to the algorithm. Then the algorithm assembles (either deterministically or randomly) its combined list. After that some item is requested. If it is not in the chosen combined list, then the algorithm incurs a miss (one unit of loss). Some or all of the base lists might also miss the requested item and might update themselves accordingly for the next trial.

The goal of the algorithm is to endure small additional loss (regret) over the best combined list chosen in hindsight once the entire request sequence and the time-varying contents of the base lists are known. We seek efficient online algorithms that implicitly maintain some information about all roughly N^K combined lists and can efficiently choose or sample a combined list from this information. As we shall see, randomness is a necessary ingredient for algorithms with good regret guarantees.

☆ An extended abstract of this paper appeared in [34].

E-mail addresses: manfred@cse.ucsc.edu (M.K. Warmuth), wouter@cs.rhul.ac.uk (W.M. Koolen), dph@cse.ucsc.edu (D.P. Helmbold).

[1] Supported by NSF grant IIS-0917397 and a Google gift grant.
[2] Supported by NWO Rubicon grant 680-50-1010.

0304-3975/$ – see front matter
http://dx.doi.org/10.1016/j.tcs.2013.09.021

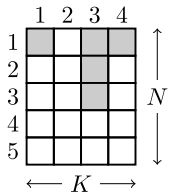

Fig. 1. We depict one combined list formed by taking the tops from $K = 4$ lists. Note that all lists have size $N = 5$ and the combined list has the same size. The list shown is $\boldsymbol{c} = (1, 0, 3, 1)$.

We claim that this setup has many applications. For example we can use our method to combine the listings produced by different search engines. Here a first goal is to combine the tops of the base lists for the purpose of maximizing hits. However in the case of search engines, we are also concerned with accumulating hits at the top of our chosen combined list and this aspect is not yet modeled by our approach. We briefly discuss such extensions in the conclusion Section 7.

Another important application is caching. In this case each list is the ranked list of items selected by a different caching strategy. We assume that all items have the same size and exactly N fit into the fast memory. The algorithm can simulate K caching strategies as "virtual caches" and then combines these virtual caches to form one "real cache". The virtual caches only record a few bytes of meta-data about each item in their cache: ID, link to the data, and calculated priority. Object data is only kept for the N items of the real combined cache. The memory cost for maintaining the virtual caches is negligible. The real combined cache can be updated as the virtual caching strategies change their item lists and the algorithm observes the performance of its combined cache.

Previous work. Methods for building a real cache by combining a number of virtually maintained caching strategies using exponential weights were explored in [15]. The employed heuristics performed well experimentally. However no performance bounds were proven. The idea for building a real cache by combining the tops of two virtual lists was first developed in [26,27]. In this case the first list contained the most recently requested items and the second contained those that were requested more than once. The goal for building a combined list was to optimally balance recency and frequency information. A deterministic heuristic was developed for adjusting the top portion taken from each list. In this paper we prove a lower bound for any deterministic algorithm that shows that any such algorithm can be forced to have loss at least KB where B is the loss of the optimal combined list chosen in hindsight. We also give a randomized online algorithm for the disjoint case whose expected loss is at most B plus an additional sublinear regret and show that the regret of this algorithm is optimal within a factor of $\min\{\sqrt{K}, \sqrt{\ln(N/K)}\}$.

This paper was motivated by our previous work on combining caching heuristics [15]. In general, there are always two approaches to a rich problem setting. Either we can restrict the setting enough so that theoretical bounds are obtainable, or we can explore heuristics without provable guarantees that perform well in practice. Ideally the two approaches will meet eventually. In this paper we focus on algorithms for which we can prove regret bounds and we believe we made significant progress towards addressing practically interesting problems.

Aggregating experts. One of the main metaphors of online learning has been the aggregation of many simple "experts" [25, 32,9]. In our application, each combined list serves as an expert. The online algorithm maintains its uncertainty over all experts as a mixture distribution and predicts based on this mixture. In more detail, the probability of a combined list $\boldsymbol{c}$ is proportional to $\beta^{\boldsymbol{M}(\boldsymbol{c})}$, where $\boldsymbol{M}(\boldsymbol{c})$ is the number of misses of list $\boldsymbol{c}$ and the update factor β lies in $[0, 1)$.

There are exponentially many mixture weights (one per expert). However, as we shall see later, we still manage to obtain very efficient algorithms by manipulating the weights implicitly. This paper is the next installment in a rich tradition of papers exploiting the combinatorial structure of the chosen comparator class (see references in [23]). In our case of combined lists, we were able to augment dynamic programming [30] with an additional speedup using Fast Fourier Transforms.

We first discuss how to obtain a combined list from a mixture. One could consider thresholding the mean, using the median (introduced below, but only works for 2 lists) or sampling.

The weighted average (mean) does not correspond to a combined list. A deterministic algorithm would naturally predict with the weighted majority (mean) of the mixture. That is, an item is in the chosen combined list if the total weight of all combined lists in the mixture that contain this item is at least 50%. Unfortunately, the weighted majority of the mixture does not always correspond to a list of size N. For $N = 4$ and $K = 3$, consider the case shown in Fig. 2, where we mix 3 combined lists each containing the top halves of 2 out of the 3 base lists. With the uniform mixture of these 3 combined lists, 6 items have total weight 2/3, more than the $N = 4$ allowed.

Deterministic algorithm for the case of two lists based on the median. In the case of $K = 2$ (i.e. two lists of size N), there are $N + 1$ combined lists $\boldsymbol{c}_0, \ldots, \boldsymbol{c}_N$, where $\boldsymbol{c}_i = (i, N - i)$ contains the i top elements from the first list and the $N - i$ top elements from the second list. These combined lists are displayed in Fig. 3.